VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Digitální VST syntetizátor

Digital VST synthesizer

2013 **Jan Hyvnar**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Hyvnar**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Digitální VST syntetizátor
Digital VST Synthesizer

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat digitální VST syntetizátor.

1. Nastudujte teorii digitální syntézy zvuku a obecně popište její principy.
2. Vyberte jeden způsob generování zvuku v reálném čase dle vlastního výběru (např. aditivní syntéza, frekvenční modulace, fázová modulace) a ten popište.
3. Nastudujte problematiku VST plug-inů a popište vzájemné vazby mezi VST hostem a VST klientem.
4. Naimplementujte jednoduchý program pro generování zvuku v reálném čase s grafickým uživatelským rozhraním a možností importovat jej jako VST plugin pro hostující sekvencery umožňující import VST pluginů. Celý váš postup popište.

Seznam doporučené odborné literatury:

[1] Users' Guide to Sound Synthesis with VST Instruments, Millward S., ISBN: 1929685785, 2002.
[2] Sound Synthesis with VST Instrument, Millward S., ISBN: 1870775732, 2002.
[3] VST - A Journey Worth pursuing, Lewis J., ISBN: 1435736990, 2008.
[4] The Audio Programming Book, Boulanger R., ISBN: 0262014467, 2010.
[5] Designing Sound, Farnell A., ISBN: 0262014416, 2010.
[6] Sound design, Teocharisová V., ISBN: 9788086253534, 2009.
[7] The Theory and Technique of Electronic Music, Puckette M.,ISBN: 9812700773, 2007.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Vích**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 6. 5. 2013

.....
~~podpis studenta~~

Poděkování

Rád bych poděkoval Ing. Lukášovi Víchovi za odbornou pomoc, rychlou reakci na dotazy a konzultaci při vytváření této diplomové práce.

Abstrakt

Tato práce se zabývá způsoby digitální syntézy zvuku, s jednotlivými metodami, se zařízeními jí realizující a její implementace pomocí technologie VST firmy Steinberg. Cílem je seznámit čtenáře s obecnou strukturou VST pluginů, uvést jej do dané problematiky a vysvětlit všechny důležité aspekty spjaté s jejich implementací. Výklad je zaměřen na nejnovější verzi VST koncepce označované jako VST3. Hlavním cílem je implementovat a představit implementaci jednoduchého VST pluginu s grafickým uživatelským rozhraním. Tento plugin využívá metodu frekvenční modulace, která je v této práci také popsána.

Klíčová slova

VST plugin, frekvenční modulace, digitální syntéza, digitální zpracování zvuku, syntezátor, C++, Steinberg

Abstract

This work describes the ways of digital sound synthesis, their various principles, devices realizing them and their implementation based on technology VST by Steinberg Company. The target is introduce reader with common structure of VST plugins, bring him to the existent problems and explain all of the important aspects linked with their implementation. Explanation is focused to the latest version of VST conception known as VST3. The main goal is implementation and presentation of simple VST plugin with graphical user interface. This plugin uses frequency modulation synthesis, which is also described in this work.

Key words

VST plugin, frequency modulation, digital synthesis, digital sound processing, synthesizer, Steinberg

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
ADSR	Attack-Decay-Sustain-Release	Vzestup-Pokles-Trvání-Uvolnění
AM	Amplitude modulation	Amplitudová modulace
DFT	Discrete Fourier transformation	Diskrétní Fourierova transformace
DSP	Digital signal processing	Digitální zpracování signálů
FIR	Finite impulse response	Filtr s konečnou impulzní odezvou
FM	Frequency modulation	Frekvenční modulace
IIR	Infinite impulse response	Filtr s nekonečnou impulzní odezvou
LPCM	Linear pulse-code modulation	Lineární pulzně kódová modulace
MIDI	Musical Instrument Digital Interface	Rozhraní pro digitální hudební nástroje
PCM	Pulse-code modulation	Pulzně kódová modulace
VST	Virtual Studio Technology	Technologie virtuálního studia
FFT	Fast Fourier transformation	Rychlá Fourierova transformace

Obsah

1 Úvod

Cílem bakalářské práce je implementovat digitální VST syntezátor a shrnout možnosti implementace VST pluginů. VST je poměrně nová technologie, která umožňuje přenést činnost klasických digitálních syntezátorů do osobních počítačů a integrovat jejich funkci s programy pro tvorbu hudby. Technologie je široce používaná v audiovisuálním průmyslu – každý, kdo si koupil CD současných interpretů, nebo zašel do kina na aktuální film, s technologií VST se setkal.

Obsah bakalářské práce, zejména část s implementací pluginů, mohou ocenit programátoři, kteří se zajímají o tvorbu VST pluginů. V současnosti totiž neexistuje žádná publikace, která by shrnovala základy pro tvorbu pluginů.

Práce je rozdělena na pět částí. V první části je vysvětlena podstata zvuku jako takového a jeho digitální reprezentace. Druhá část popisuje obecné principy v rámci digitální syntézy, včetně popisu nejčastějších způsobů syntézy a nejpoužívanějších DSP algoritmů. V části je také rozebrán klasický syntezátor a popsány jeho části.

Třetí část se věnuje frekvenční modulaci, jednomu z možných způsobů digitální syntézy, který je aplikován v implementované aplikaci FM Synth. Je také rozebrána historie frekvenční modulace a popsán její princip.

Čtvrtá část rozebírá VST plugin z jeho technické stránky. Část je zaměřena na nejnovější verzi VST3. Je rozebrána základní struktura VST3 pluginu a základní informace potřebné k implementaci pluginu.

Poslední část podrobně rozebírá praktickou část bakalářské práce – implementaci pluginu FM Synth založeném na principu frekvenční modulace. V části jsou popsány další možné techniky tvorby VST pluginů.

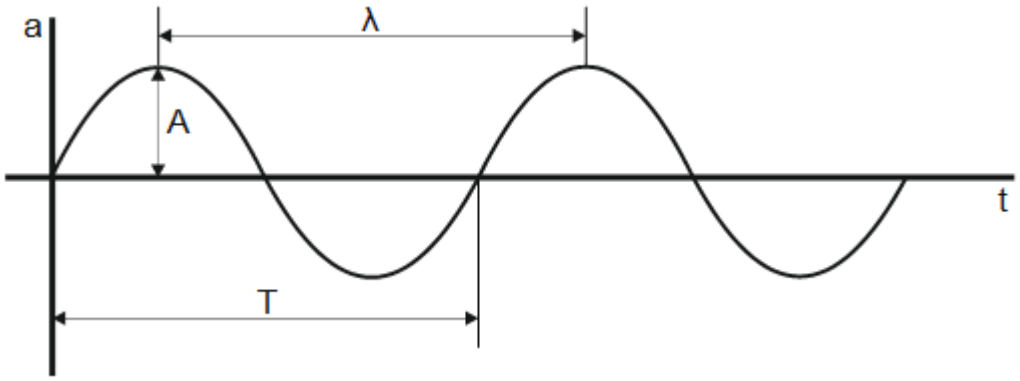
2 Zvuk

V kapitole jsou použity informační zdroje [1] až [4].

Zvuk je mechanické vlnění v látkovém prostředí, které vnímáme uchem, a v mozku vyvolává zvukový vjem. Kmitočtový rozsah, který dokážeme vnímat je individuální. Obvykle se jedná o hodnoty mezi 16 Hz až 20 kHz. Vlnění mimo tento rozsah zvukový vjem u člověka nevyvolá, nicméně frekvence pod 16 Hz, obecně nazývány jako infrazvuk, vnímají například sloni, frekvence nad 20 kHz, označované jako ultrazvuk, vnímají například delfíni, psi a netopýři. Vědní obor, který zkoumá děje spojené se vznikem, šířením a vnímáním, se nazývá akustika.

2.1 Vznik zvuku

Zvuk vzniká kmitáním bodů a bodových soustav. Kmitavý pohyb je periodická změna nějaké veličiny (poloha, rozměr, tlak, rychlost,…) v čase. Zdrojem zvuku může tedy být jakékoliv chvějící se těleso. O vlnění však nerozhoduje pouze toto chvění, ale také okolnost, zda je předmět dobrým nebo špatným zářičem zvuku. Zařízení, které vykonává kmitající pohyb, se nazývá oscilátor.



Obrázek 2.1 - Parametry obecného harmonického průběhu

Pohyb je charakterizován:

- **Periodou *T*** – čas, za který soustava udělá jeden kmit. V akustice se většinou používá počet period za jednotku času (tedy za 1 sekundu) – veličina se označuje jako frekvence, její jednotkou je Hertz (Hz) a vztah mezi periodou uveden ve vzorci 2.1.

$$f = \frac{1}{T} \tag{2.1}$$

- **Maximální výchylkou (amplitudou) *A*** – největší vzdálenost, o kterou se soustava vychýlí od rovnovážné polohy.
- **Okamžitou amplitudou *a*** – vzdálenost soustavy od rovnovážné polohy ve sledovaném okamžiku *t*.

- **Vlnovou délkou λ** - vzdálenost dvou nejbližších bodů v rámci vlnění, které mají totožnou okamžitou amplitudu a. Vztah mezi frekvencí a vlnou délkou je zobrazen ve vzorci 2.2.

$$\lambda = \frac{v}{f} \tag{2.2}$$

v je rychlost šíření vlnění (u zvuku *v = 345 m/s*)

S použitím těchto vlastností můžeme matematicky zapsat průběh harmonického signálu, který je uveden ve vzorci 2.3.

$$a = A \sin\left(\frac{2\pi}{T} t\right) = A \sin(2\pi f t) = A \sin(\omega t) \tag{2.3}$$

Vodičem pro přenos zvuku je nejčastěji vzduch. Zvuky se mohou šířit také kapaliny či pevné látky, ale v případě těchto vodičů dochází ke zkreslení zvuku. Vakuum je zvukovou izolací.

Během toho, jak je generován vzduch zdrojem zvuku se částice zvuku v některých místech prostoru navzájem přibližují či vzdalují, tím vzniká jejich zhuštění nebo zředění (přetlak, podtlak). Zejména při šíření prostorem s různými teplotami vzduchu. Pokud převedeme tuto změnu rychlosti na změnu výšku tónu, může se nástroj samovolně přeladit až o jeden půltón.

2.2 Rozdělení zvuku

Veškeré zvuky vlny se obecně rozdělují na tón a na hluk.

Tón je signál se stále stejnou frekvencí. Tóny se dále dělí do dvou skupin – na tony jednoduché, jako je například funkce sinus a tóny složené, jejichž průběh je periodický, ale obsahují kromě základní frekvence také i tzv. vyšší harmonické.

Každý tón má tyto parametry:

1. **Výška** – výška je dána frekvencí tónu, čím je vyšší frekvence, tím je vyšší tón. U jednoduchého tónu se určuje absolutní výška tónu, protože se jedná o prostý harmonický průběh a ten má pouze jednu frekvenci. U složených tónu se jako absolutní výška tónu uvádí frekvence, která je násobkem pro všechny ostatní frekvence.

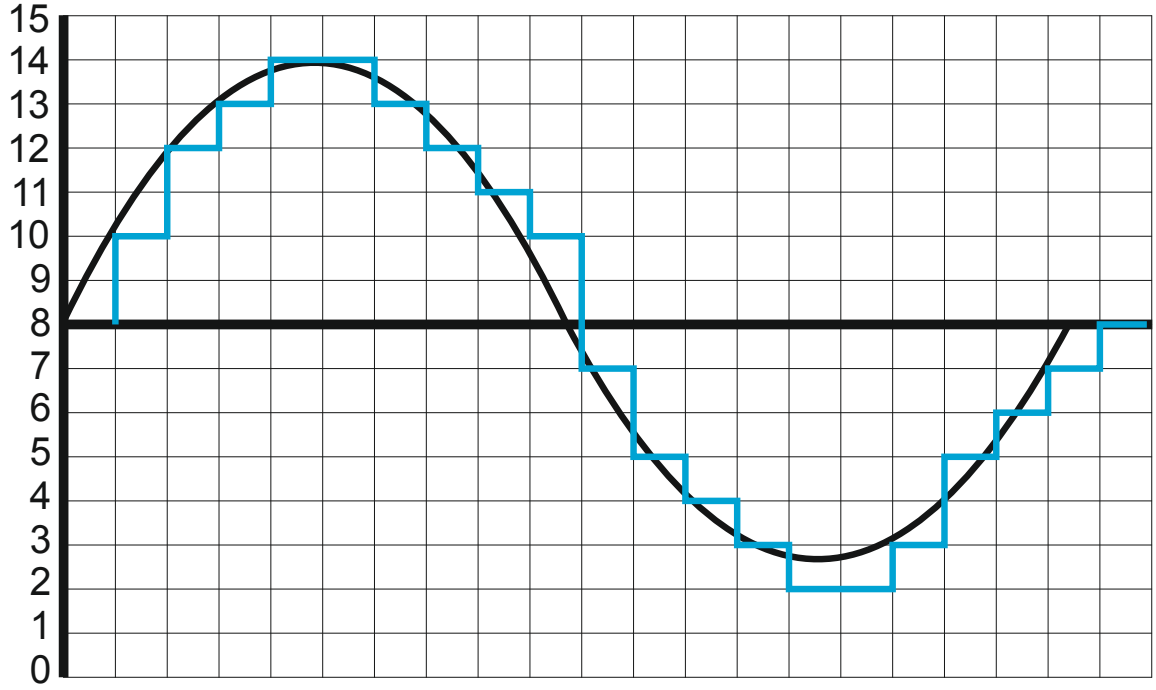
2. **Barva** – každý tón může mít pro naše ucho zcela charakteristický zvuk, což je dáno právě obsahem vyšších harmonických tónu ve složeném tónu.

3. **Hlasitost zvuku** – hlasitost je subjektivní veličina. Obecně je definována jako dvacetinásobek desítkového logaritmu, jehož argumentem je poměr mezi velikostí akustického tlaku a vůči smluvené vztažné hodnotě akustického tlaku, často označována jako práh slyšení.

Naproti tomu hluk jsou nepravidelné vlnění vznikající jako složité nepravidelné kmitání těles, nebo krátké nepravidelné rozruchy. I hluky jsou využívány v hudbě, neboť k nim patří i zvuky mnoha hudebních nástrojů, například bicí.

2.3 Digitální reprezentace zvuku

Digitální zvuk je obvykle reprezentován pomocí lineární pulzně kódové modulace. Signál reprezentován pulzně kódovou modulací má dva základní parametry – vzorkovací frekvenci (počet vzorků za sekundu) a rozlišení v bitech, které definuje počet možných úrovní.



Obrázek 2.2 - Digitální reprezentace zvuku

Sinusovka je navzorkována v pravidelných intervalech (sloupce) a pro každý vzorek je přiřazena jedna z možných hodnot (řádky) v závislosti na použitém algoritmu. Tento proces se nazývá kvantování, je vždy ztrátový a nevratný. Vzorkovací frekvence by vždy měla splňovat Nyquistovy vzorkovací věty (Shannon-Kotělnikův teorém) – musí být více, než dvojnásobná oproti zaznamenávané frekvenci. V případě, že by tato podmínka splněna nebyla, docházelo by k aliasingu. Totéž platí pro rozlišení. Při nižším rozlišení je velice problematické signál co nejvěrněji zrekonstruovat. Obvykle se digitálně uchovaný zvuk používá rozlišení 16 bitů (65 536 možných stavů) a vzorkovací frekvence 44 100 Hz.

3 Digitální syntéza zvuku

V kapitole jsou použity informace ze zdrojů [5] až [18]

Digitální syntéza zvuku jsou principy pro generování zvuku s využitím digitálních metod. Přístroj, který je realizuje, se nazývá syntezátor.

3.1 Počátky digitální syntézy

První pokusy o zvukovou syntézu realizovanou počítačem začaly v roce 1957 v Bellových laboratořích. Skupina výzkumníku v čele s Maxem Mathewsem vyvinula počítačový program a programovací jazyk pro sálový počítač IBM 704 s názvem Music I. - ten však umožňoval vygenerovat pouze jeden tón s tím, že uživatel může naprogramovat jeho výšku a délku. Vygenerovaný zvuk byl naprosto přesný.

O rok později byla vyvinuta další verze Music II. Tato verze využívala počítač IBM 7094, který na rozdíl od IBM 704 používal tranzistory, což zvýšilo možnosti digitální syntézy. Díky tomu byla k dispozici polyfonie (až čtyři hlasy) s možností šestnácti průběhů uložených v paměti.

Třetí verze byla uvolněna v roce 1960. Ta přinesla koncept uživatelských generátorů, což jsou funkční bloky jako oscilátory, filtry a zesilovače, které mohou být navzájem propojeny a které generují barvu tónu definovanou uživatelem.

Vývoj řady Music-N pokračoval i nadále. Všechny další verze čerpaly z konceptu uživatelských generátorů. Music IV. byl v podstatě Music III. přepsán do jazyka symbolických adres s využitím maker, který se nazýval BAFEP. Music IV. umožňoval vložení notové osnovy jako textového souboru. Music V. umožňoval také zpracovávat digitalizovaný signál.

Všechny Music-N využívají tzv. Wavetable syntézu – generování zvuku neprobíhá v reálném čase, ale vzorky jsou již uložené v paměti. V případě Music I. a II. jsou tyto průběhy uloženy v paměti, u Music III. až V. může programátor, díky konceptu uživatelských generátorů, vytvářet vlastní průběhy.

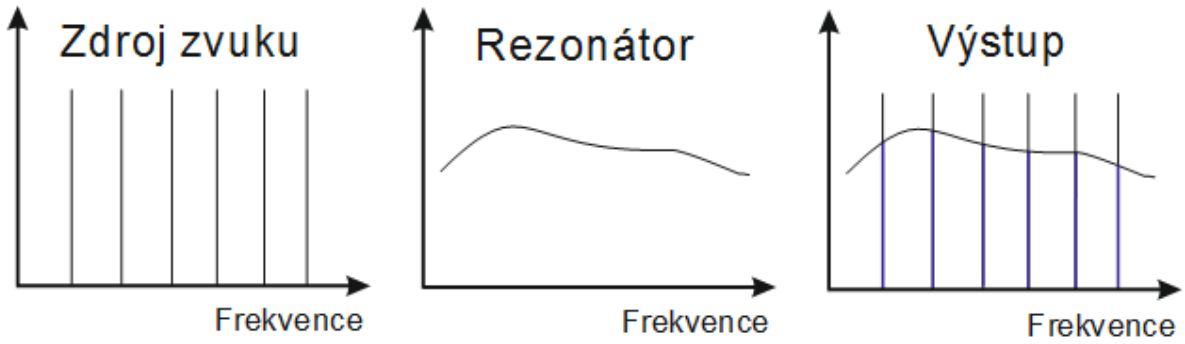
3.2 Způsoby digitální syntézy

Existuje několik možných způsobů, jak realizovat digitální syntézu. V této kapitole jsou rozebrány nejpoužívanější a nejefektivnější z nich.

3.2.1 Rozdílová syntéza

Rozdílová syntéza je druh syntézy, která je nejpoužívanější u většiny komerčních syntezátorů. Hlavní výhodou je jednoduchost. Fyzikální princip rozdílové syntézy je analogický s přirozeným vznikem zvuků v hudebních nástrojích nebo v tvorbě lidského hlasu.

Během rozdílové syntézy nevznikají žádné nové složky zvuku, stávající jsou buď potlačeny, nebo zesíleny. To je značné omezení možností pro generování zvuku, proto obvykle tento druh syntézy bývá doplňován některým z dalších způsobů (například frekvenční modulace).



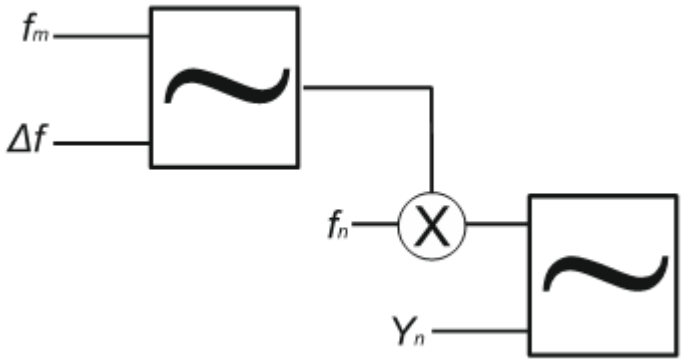
Obrázek 3.1 - Základní princip rozdílové syntézy

Příkladem rozdílové syntézy z reálného světa může být hra na housle – zdroj (spektrum) zvuku (kmitající struna) je tvarováno korpusem houslí.

Většina syntezátorů již má vygenerované základní vzorky a ty uživatel může různě měnit právě díky nastavení rezonátoru, který určuje výslednou barvu tónu.

3.2.2 **Frekvenční modulace**

Barva zvuku je tvořena průběhem funkce, jejíž frekvence je modulována nějakým dalším průběhem. Princip byl patentován v roce 1975, stal ve velice populárním v osmdesátých letech díky syntezátorům firmy Yamaha DX7.



Obrázek 3.2 – Blokové schéma syntézy pomocí frekvenční modulace

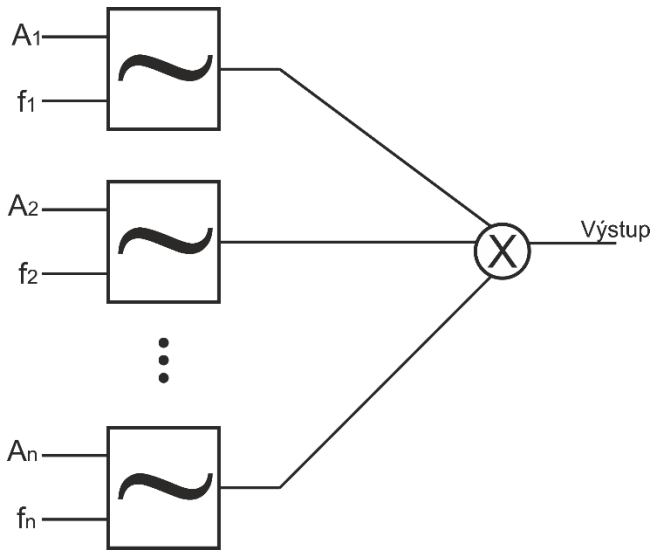
Během frekvenční modulace vzniká velké množství dalších kmitočtů. Tyto kmitočty jsou obvykle úměrné součtům a rozdílům již přítomných frekvencí. Jediná možnost, jak lze nepřímo vzniklé kmitočtové pásmo ovlivnit, je skrze modulační index – což je poměr mezi rozdílem frekvence modulační od frekvence nosné a frekvencí modulační.

$$y(t) = Y_N \sin(2\,\pi\, f_N\, t + \beta\ \sin(2\,\pi\, f_M\, t)\, t\,) \qquad (4.4)$$

Frekvenční modulaci je věnována kapitola č. 4.

3.2.3 **Aditivní syntéza**

Aditivní syntéza vytváří výslednou barvu tonu pomocí skládání několika harmonických průběhů. Na rozdíl od frekvenční modulace můžeme přesně určit, které kmitočty se budou ve frekvenčním spektru vyskytovat.



Obrázek 3.3 – Princip aditivní syntézy

Princip aditivní syntézy vychází z Fourierovy transformace.

$$y(t) = \sum_{k=0}^{K} Y_k(2\,\pi\,k\,f_0\,t\,+\,\varphi) \tag{3.2}$$

Vzorec 3.2 pro výpočet hodnoty aditivní syntézy je v tzv. harmonické formě, protože skládané frekvence by měly být harmonické, v jiném případě by výsledný zvuk nezněl příjemně pro lidské ucho. Frekvence může být rozlaďována pomocí fázového posunu.

Aditivní syntéza je použita ve velké řadě syntezátoru, včetně těch nejstarších, které nebyly ještě digitální – například u Hammondových varhan. Své využití nachází i v syntéze řeči.

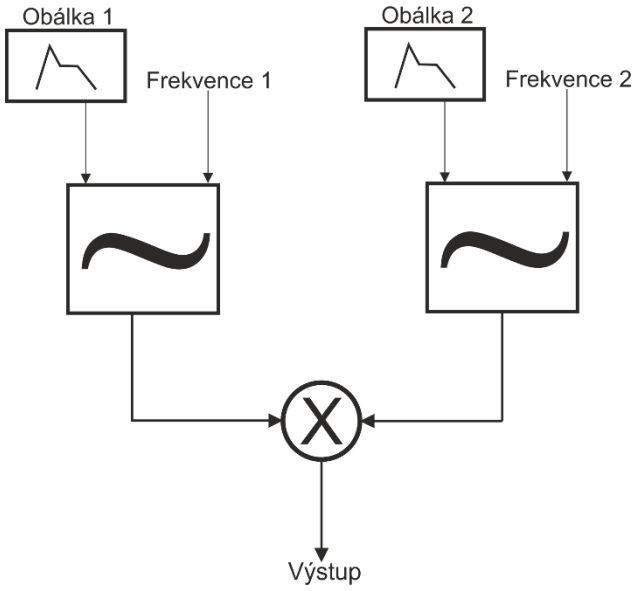
3.2.4 **Syntéza uložených vzorků**

Syntéza uložených vzorků je naprosto odlišná od všech ostatních. Rozdíl je v tom, že veškerý zvuk je již zaznamenán (navzorkován) a ten se pouze přehrává, či nějak upravuje. Evidentní výhodou tohoto principu je nižší nároky na vypočtenost (nic se nemusí generovat), nicméně často nastávají situace, kdy je třeba přehrát jeden záznam vícekrát, než jednou a kdy musí být zahráno současně více záznamů. První nástroj, který syntézu uložených vzorků realizoval, byl mellotron, který však pracoval ještě s analogovými vzorky.

První čistě digitální syntezátory založené na tomto principu měly v paměti uloženy pouze jeden kmitočet a ten byl zrychlován nebo zpomalován k dosažení požadované frekvence. To bylo v době, kdy finanční náklady na paměti byly vysoké. V současnosti může mít nejen každá nota svůj vzorek, ale také více vzorků pro každý tón (například pro různé techniky).

3.2.5 **Wavetable syntéza**

Wavetable syntéza je podobná aditivní nebo syntéze uložených vzorků, liší se zejména ve dvou věcech – zaprvé tabulka se vzorky neobsahuje jen jednu periodu signálu sinusového signálu, ale jednu periodu signálu ze sinusových signálů modifikovaných do nového tvaru vlny. Za druhé existují funkcionality pro dynamickou změnu tvaru vlny, její amplitudy nebo obálky vlny.

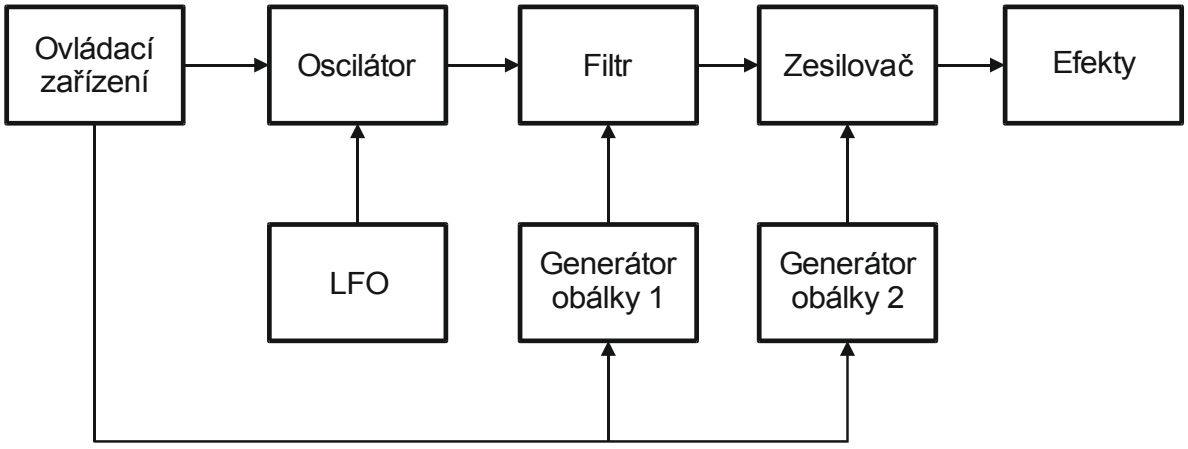


Obrázek 3.4 – Princip wavetable syntézy

Mezi nejznámější syntezátory používající tuto techniku patří například D-50 od firmy Roland.

3.3 **Syntezátor**

Syntezátor je elektrický přístroj, který generuje zvukový signál a realizuje jeden a více z výše uvedených způsobů digitální syntézy.



Obrázek 3.5 – Blokové schéma syntezátoru

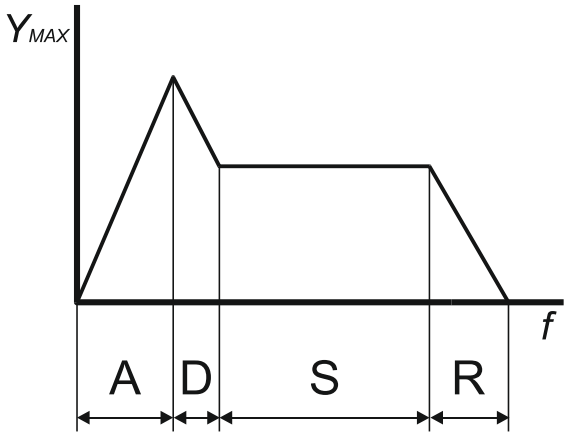
Syntezátor může obsahovat tyto komponenty:

- **Ovládací zařízení –** ovládací zařízení určuje, který tón bude přehrán. Většina syntezátorů používá různě velkou klaviaturu, obvykle se čtyřmi oktávami. Téměř vždy je možné zvuk dále transponovat.
- **Oscilátor –** oscilátor je srdcem syntezátoru. Vytváří zvuky v závislosti na použitém typu syntézy – aditivní syntéza používá *n* oscilátorů, jejichž výstupy sečte; frekvenční modulace používá dva oscilátory, přičemž jeden navzájem moduluje druhý apod. Nejčastěji se používají napěťově či digitálně řízené oscilátory.
- **Filtr** – tvaruje oscilátory vygenerovaný zvuk – propouštějí frekvence či část vygenerovaného signálu. Používají se horní, dolní či pásmové propusti.
- **Napěťově řízený zesilovač** – po všech předchozích úpravách je zvuk zesilovačem zesílen. Jedná se o předzesilovač, signál bude zesílen znovu s ohledem na nastavenou hlasitost.
- **Generátor nízkofrekvenčních signálů (LFO)** – oscilátor, který generuje zvuky nastavené frekvence, který rytmicky modulují generovaný signál – zejména pro efekty, jako tremolo, vibrato a podobně.
- **Generátory obálek** – běžně syntezátor obsahuje dva generátory obálek. První z nich řídí ostrost zvuku (brightness) a druhý tvaruje výsledný signál do požadované obálky (ADSR)
- **Další efekty** – syntezátor může obsahovat různou škálu dalších efektů, jako je chorus, delay, apod.

Filtr se používá zejména u rozdílové syntézy, nicméně často se používá také jako jeden z efektů. Přináší totiž další možnosti, jak modifikovat vygenerovaný signál. Může tak například eliminovat, nebo zesílit některé frekvence vzniklé během FM syntézy.

3.3.1 Obálka

Většina hudebních nástrojů negeneruje po dobu trvání zvuku stejný akustický tón. Ten obvykle začíná úderem a postupně zní do ztracena. U syntezátorů se nejčastěji používá ADSR obálka, která je rozděluje tón na čtyři části: Náběh, pokles, trvání a doznívání.



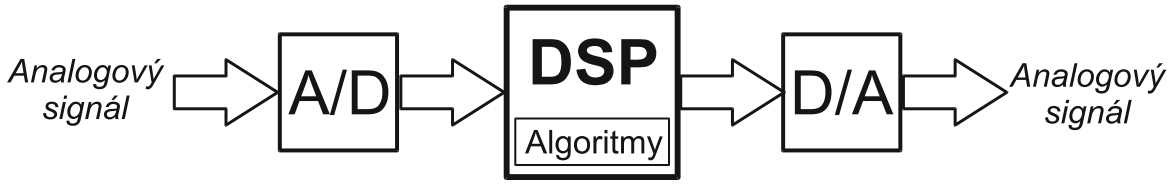
Obrázek 3.6 – Obálka

- **Attack (náběh)** – doba, za kterou signál vzroste z nuly na maximální hodnotu (hlasitost). Tento parametr značně ovlivňuje barvu tónu.
- **Decay (pokles)** – doba, za kterou signál klesne z maximální hodnoty (hlasitosti) na úroveň fáze trvání.
- **Sustain (trvání)** – doba, jak dlouho se bude signál opakovat, než bude klávesa uvolněna. U většiny syntezátorů je obvykle tato část nastavena na hodnotu nekonečno. V případě, konkrétní hodnoty se signál opakuje určenou dobu a poté automaticky nastane fáze doznívání.
- **Release (doznívání)** – doba, za jak dlouho hlasitost po skončení doby trvání, nebo po uvolnění klávesy.

U pokročilejších syntezátorů se můžeme setkat s více krokovou obálkou – pokles a trvání mohou různě klesat a zase stoupat. Syntezátory na úrovni počítačových programů mohou generovat obálky s neomezeným počtem kroků. Jednotlivé klesání a stoupání signálu v rámci ADSR obálky nemusejí být pouze lineární – mohou být exponenciální, logaritmické apod.

3.4 Digitální signálový procesor

Digitální signálový procesor je mikroprocesor optimalizovaný pro zpracování digitálně reprezentovaných signálů. Tvoří srdce většiny digitálních syntezátorů.



Obrázek 3.7 – Blokové schéma DSP procesoru

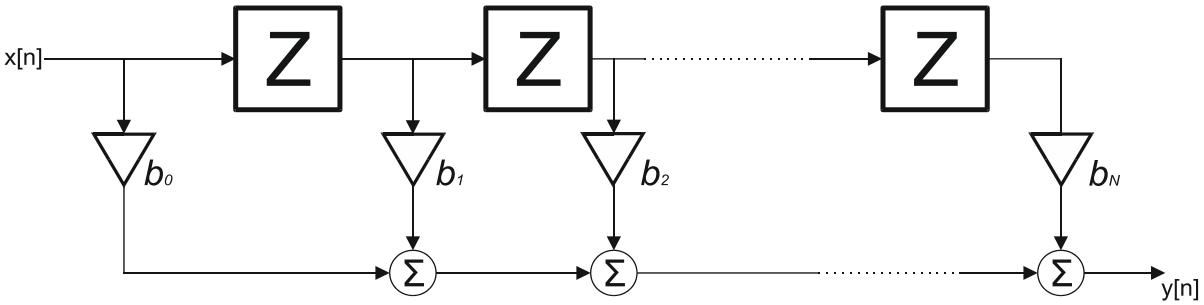
Na obrázku 3.7 je zobrazena typická koncepce DSP procesoru. Analogový signál je převeden na digitální, ten je upraven a následně převeden zpět na digitální. Většina syntezátorů pomocí digitálních oscilátorů generuje digitální vstupní signál, tudíž nutnost předvádět tento signál odpadá.

Samotný DSP procesor využívá harvardskou strukturu. Výpočty jsou zrychleny tím, že obsahuje speciální jednotky, které dokáži pracovat paralelně. Klasický DSP procesor má také rychlou násobičku, která se používá u většiny algoritmů zpracování digitálního signálu. Mezi nejčastěji použitelné DSP algoritmy patří:

- Filtr s konečnou impulzní odezvou
- Filtr s nekonečnou impulzní odezvou
- Fourierova transformace

3.4.1 **Filtr s konečnou impulzní odezvou**

Filtr s konečnou impulzní odezvou je filtr, jehož impulzní odezva (tj. odpověď na jakýkoliv konečný vstup) má konečné trvání, protože se usadí na nule v konečném čase.



Obrázek 3.8 – Blokové schéma FIR filtru

Na obrázku 3.8 je zobrazeno blokové schéma FIR filtru. *Z* značí zpožďovací obvod, b_0-b_N koeficienty, *x[n]* je vstup, *y[n]* je výstup, *N* je řád filtru. Každý filtr má navrženy koeficienty a řád filtru nastavený tak, abych dosáhl požadovaného pásma a tudíž tvaru vlny.

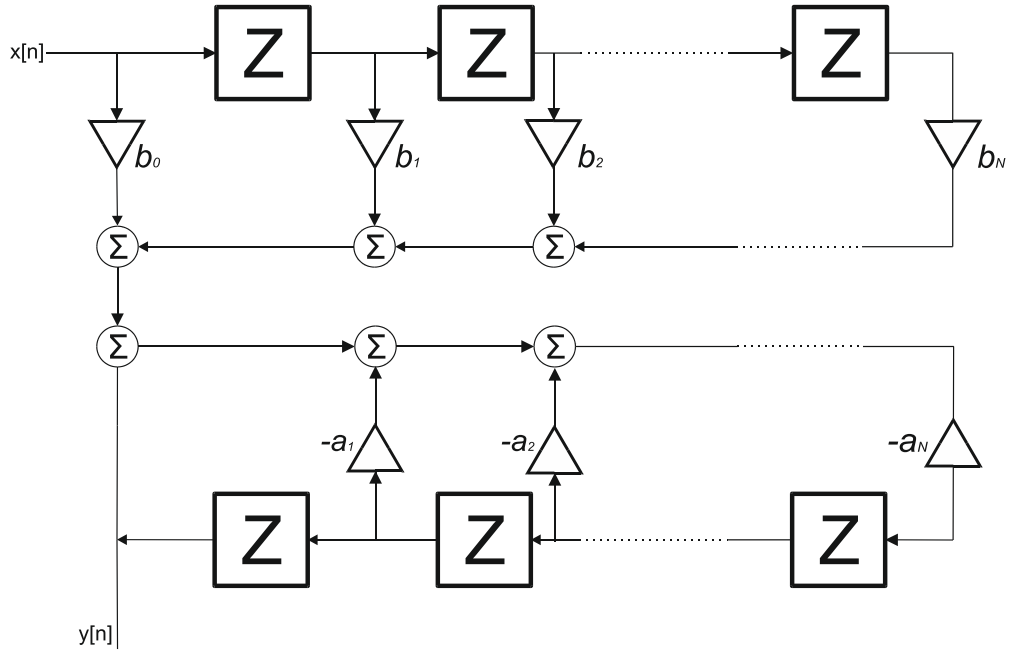
$$y(t) = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] = \sum_{i=0}^{N} b_i x[n-i] \qquad (3.3)$$

FIR filtr nepotřebuje žádnou zpětnou vazbu. Díky tomu jsou FIR filtry vždy stabilní – už jenom proto, že celý tok dat tvoří kruh (což je podmínka pro stabilitu v diskrétních, časově lineárních invariantních systémů).

FIR filtry mohou být velmi snadno navrženy s lineární fází – koeficienty budou lineární. Lineární fáze či fázový posun jsou přímo úměrné frekvenci, což způsobuje konstantní zpoždění na všech frekvencích. To se používá v systémech, kde jsou kladeny požadavky na přesnost fáze – například crossover filtry nebo mastering.

3.4.2 **Filtr s nekonečnou impulzní odezvou**

IIR filtry mají nekonečnou impulzní odezvu a vyžadují minimálně jednu zpětnovazební smyčku. Přenos je tvořen podílem polynomů – řád filtru je určen nejvyšším stupněm polynomu.



Obrázek 3.8 – Blokové schéma IIR filtru

Z obrázku 3.8 vyplývá tento matematický vztah pro závislost mezi vstupem a výstupem, který je uveden v rovnici 3.4.

$$\begin{aligned} y(t) = {} & b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] - a_1 x[n] \\ & + a_2 x[n-1] + \cdots + a_N x[n-M] \\ & = \sum_{i=0}^{N} b_i x[n-i] \; - \sum_{j=1}^{M} a_j x[n-j] \end{aligned} \tag{3.4}$$

IIR filtry ve srovnání s FIR filtry používá pro svoje mnohem menší množství paměti. Nevýhodou je, že v důsledku zpětných vazeb je zde větší náchylnost k saturaci aritmetiky procesoru.

3.4.3 Fourierova transformace

Fourierova transformace je matematický aparát pro vyjádření hodnoty funkce v čase jako harmonických signálů (funkce sinus a cosinus). Funkce převádí signál z časové oblasti do frekvenční. Rozlišujeme Fourierovu transformaci ve spojitém a nespojitém čase. V DSP se tedy využívá Diskrétní Fourierova transformace.

$$X(t) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j\pi nm}{N}} = \sum_{n=0}^{N-1} \left[\cos\left(\frac{2\pi nm}{N}\right) - j sin\left(\frac{2\pi nm}{N}\right) \right] \tag{3.5}$$

$$x(t) = \frac{1}{N}\sum_{n=0}^{N-1} X(n)e^{-\frac{j\pi nm}{N}} = \frac{1}{N}\sum_{n=0}^{N-1}\left[\cos\left(\frac{2\pi nm}{N}\right) - jsin\left(\frac{2\pi nm}{N}\right)\right] \qquad (3.6)$$

Rovnice 3.5 popisuje diskrétní Fourierovu transformaci a rovnice 3.6 její inverzní tvar.

Výpočet podle uvedeného definičního vztahu vyžaduje N^2 komplexních součinů a N^2 komplexních součtů. Tohle množství operací neumožňuje provádění DFT v reálném čase. To se změnilo až s vývojem tzv. rychlé Fourierovy transformace. FFT je definován se složitostí *O(N log N)* operací, algoritmy většinou využívají faktorizaci *N*.

Mezi nejstarší FFT algoritmus patří Cooley-Tukey algoritmus. Je implementován na bázi metody „rozděl a panuj“, který rekurzivně dělí DFT s velikostí složeného čísla do menších DFT o velikostech N_1 a N_2.

4 Frekvenční modulace

Kapitola vychází z informačních zdrojů [19] – [21].

Modulace je obecně nelineární proces, pomocí kterého se mění charakter vhodného nosného signálu pomocí modulačního signálu. Většinou se používá v radiotechnice, ale je také jedním z možných principů syntézy zvuku. Aby se tento jev projevil, musí být frekvence modulátoru ve slyšitelném spektru. Během modulace vznikají tzv. postranní pásma (tzv. modulační produkty), což jsou nové frekvence s různými amplitudami ve frekvenčním spektru se nacházející okolo nosného signálu.

4.1 Amplitudová vs. Frekvenční modulace

Existuje několik různých typů modulace. Nejběžněji používané jsou modulace amplitudová a frekvenční. Amplitudová, jak plyne z jejího názvu, ovlivňuje amplitudu signálu. Žádné další parametry vlny, jako její frekvence, nebo fáze, se nemění. Jedná se o historicky první druh modulace.

Matematický popis nosné a modulační vlny je uveden ve vzorcích 4.1 a 4.2, kde y_N (nosný signál) a kde y_M (modulační signál), y_N a y_M je okamžitá hodnota sinusového signálu, Y_M a Y_N je amplituda signálu, f_N a f_M jsou frekvence signálu.

$$y_N = Y_N \, sin(2\,\pi\, f_N\, t) \tag{4.1}$$

$$y_M = Y_M \, sin(2\,\pi\, f_M\, t) \tag{4.2}$$

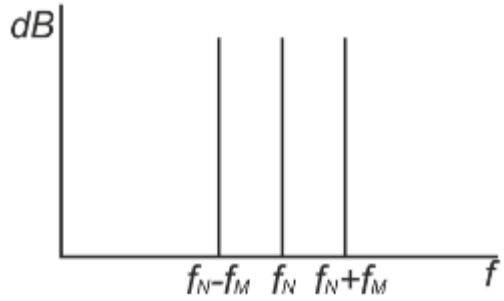
Dosadíme modulační signál do vzorce nosné tak, aby ovlivňoval amplitudu. Získáme vztah uvedený ve vzorci 4.3.

$$y(t) = (Y_N + Y_M \sin(2\,\pi\, f_M\, t)) * sin(2\,\pi\, f_N\, t) \tag{4.3}$$

Pokud tyto členy roznásobíme a pomocí goniometrických vzorců upravíme, získáme následující vztah 4.4.

$$y(t) = Y_N \sin(2\,\pi\, f_N\, t) + \frac{Y_M \cos(2\,\pi\, f_N\, t - 2\,\pi\, f_M\, t)}{2} - \frac{Y_M \cos(2\,\pi\, f_N\, t + 2\,\pi\, f_M\, t)}{2} \tag{4.4}$$

Ze vztahu uvedeného ve vzorci 4.4 vyplývá, že frekvenční spektrum amplitudové modulace bude obsahovat tyto frekvence zobrazené na obrázku 4.1.



Obrázek 4.1 – Frekvenční spektrum amplitudové modulace

Naproti tomu frekvenční modulace je mnohem složitější, zejména z toho hlediska, že se ovlivňuje přímo argument dané funkce, výsledné spektrum bude mnohem bohatší. Frekvenční modulaci stejných dvou signálů, uvedených ve vzorcích 4.1 a 4.2. Dosadíme obě rovnice tak, aby u výsledného signálu byla ovlivňována frekvence:

$$y(t) = Y_N \sin(2\,\pi\,f_N\,t + \beta\ \sin(2\,\pi\,f_M\,t)\,t\,) \qquad (4.5)$$

Během frekvenční modulace vzniká značné množství postranních pásem s různou amplitudou, která může být odvozena pomocí tzv. modulačního indexu β. Jeho význam je rozebrán níže.

4.2 Historie frekvenční modulace

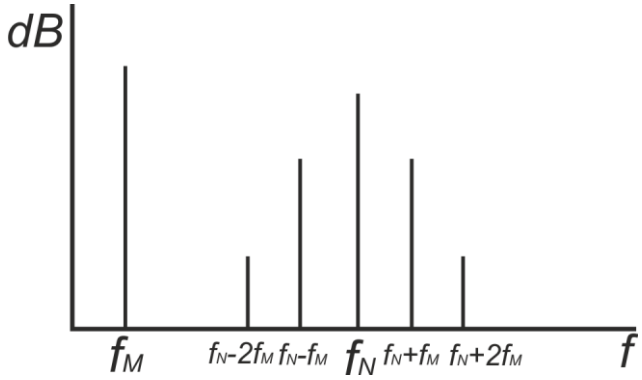
Samotná metoda frekvenční modulace byla vynalezena v 30. letech 20. století. Jejím autorem je Edwin Howard Armstrong. Její objevení znamenalo posun pro rádiovou techniku zejména z toho důvodu, že frekvenční modulace na rozdíl od AM je méně náchylná k rušení a umožňuje přenášet mnohem větší kmitočtové pásmo.

K použití FM ke zvukové syntéze došlo až o několik let později. V šedesátých letech 20. století John Chowning náhodou při výzkumu různých druhů efektu vibrato zjistil, že když frekvence modulujícího signálu dosáhne mimo určitý bod, efekt vibrato zmizí a z modulovaného tónu vznikne úplně nový tón, který naprosto nahradí ten originální. Jelikož se frekvenční modulace v té době používala v radiotechnice, tedy ve velmi vysokých frekvencích, Chowning byl tak první, kdo slyšel frekvenčně modulovaný tón. On sám zkomponoval první skladbu používající tuto techniku.

Následujících několik let strávil Chowning popisováním tohoto jevu a v roce 1964 si jej nechal patentovat. Následně pracoval na prvním komerčním syntetizátoru využívající FM – Yamaha GS1 byla uvedena na trh v roce 1981. Nutno podotknout, že tento syntetizátor byl na svou dobu značně finančně nákladný. Spolupracoval také na dalším syntetizátoru z dílny Yamahy – syntetizátor Yamaha DX7 vešel do prodeje v roce 1983.

4.3 **Postranní pásma**

Oba typy používaných typů modulací generují postranní frekvenční pásma. Jsou to dodatečné frekvence, které nemusejí být harmonické jak vůči nosné, tak vůči modulační frekvenci. Zatímco u AM jsou obsažena maximálně tři frekvenční pásma, u FM jich může být až neomezený počet.



Obrázek 4.2 – Frekvenční spektrum frekvenční modulace

$$\omega_{sb} = \omega_c \pm n\,\omega_m \tag{4.6}$$

Z obrázku 4.2 vyplývá, že pro počet postranních pásem bude platit vzorec 4.6 - každé postranní pásmo leží na frekvenci plus mínus násobku frekvence modulační posunuté o frekvenci nosné. Jelikož číslo *n* je celé reálné číslo, můžeme generovat teoreticky neomezené množství postranních pásem. Prakticky to není možné, protože žádný systém nemá nekonečné frekvenční pásmo.

Pokud se chceme zabývat amplitudami jednotlivých signálu, musíme se blíže zaměřit na tzv. modulační index, ve vzorci 4.5 označen jako *β*. K vysvětlení jeho vlivu na modulovaný signál se musíme vrátit k elementárním vzorcům časového průběhu harmonických funkcí.

$$y(t) = Y_N \sin(2\,\pi\, f_N\, t + \phi\,) = Y_N \sin(\Omega\, t + \phi\,) \tag{4.7}$$

$$\Omega(t) = \Omega + \Delta\Omega \sin(\omega\, t) \tag{4.8}$$

Y_N ve vzorci 4.7 je amplituda nosného signálu, kde Ω je úhlová frekvence nosné a ϕ značí fázový posun. Úhlová rychlost je v podstatě úhel otočení, ve kterém se oscilátor právě nachází. Abychom chtěli, aby výsledný signál byl nějakým jiným signálem ovlivňován, musí tento modulační signál ovlivňovat právě úhlovou frekvenci. Tato závislost je zapsána ve vzorci 4.8. $\Delta\Omega$ je tzv. frekvenční zdvih a ω je úhlová rychlost modulační vlny. Tento vzorec nemůžeme dosadit do vzorce 4.7, protože funkce sinus je definována pro úhly, nikoliv pro frekvenci. Proto musíme okamžitou

frekvenci vyjádřit jako úhlovou rychlost, což je změna dráhy za jednotku času. Aby mohl být úhel nalezen, musíme integrovat Ω v závislosti na čase.

$$\int \Omega \mathrm{dt} = \int (\Omega + \Delta\Omega \sin(\omega\, \mathrm{t})) = \Omega \mathrm{t} + \frac{\Delta\Omega}{\omega} \sin(\omega\, \mathrm{t}) \tag{4.9}$$

$$\begin{aligned} y_N(t) = Y_N \sin\left((\Omega + \frac{\Delta\Omega}{\omega} \sin(\omega\, \mathrm{t}))t\right) \\ = Y_N \sin\left((\Omega + \frac{\Delta \mathrm{f}}{f_N} \sin(\omega\, \mathrm{t}))t\right) \\ = Y_N \sin((\Omega + \beta \sin(\omega\, \mathrm{t}))t) \end{aligned} \tag{4.10}$$

Pomocí vzorce 4.9 nalezneme okamžitý úhel, výsledný vztah dosadíme do vzorce 4.7 s tím, že fázový posun se bude rovnat nule – je konstantní, na další výsledky nemůže mít vliv. Obdržíme potom vztah 4.10. Nyní lze vidět, že amplitudy postranních pásem jsou přímo ovlivňovány modulačním indexem. Ale ve vzorci 4.10 se nic o jednotlivých amplitudách vyčíst nelze. Vzorec může být dále rozepsán podle vzorce po součet argumentů funkce sinus.

$$y_N(t) = Y_N[\sin(\Omega \mathrm{t}) \cos(\beta \sin(\omega\, \mathrm{t})) + \cos(\Omega \mathrm{t}) \sin(\beta \sin(\omega\, \mathrm{t}))] \tag{4.11}$$

$$\cos\big(\beta \sin(\omega\, \mathrm{t})\big) = J_0(\beta) + 2J_2(\beta) \cos(2\omega \mathrm{t}) + 2J_4(\beta) \cos(4\omega\, \mathrm{t}) + \cdots \tag{4.12}$$

$$\begin{aligned} \sin\big(\beta \sin(\omega\, \mathrm{t})\big) = 2J_1(\beta) \sin(\omega \mathrm{t}) + 2J_3(\beta) \cos(3\omega \mathrm{t}) \\ + 2J_5(\beta) \sin(5\omega\, \mathrm{t}) + \cdots \end{aligned} \tag{4.13}$$

$$\sin(\mathrm{x}) \cos(y) = \frac{1}{2}[\sin(x + y) + \sin(x - y)] \tag{4.14}$$

$$\begin{aligned} y_N(t) = Y_N\{ J_0(\beta) \sin(\Omega \mathrm{t}) + J_1(\beta)[\sin(\Omega \mathrm{t} + \omega)\, \mathrm{t} \\ + \sin(\Omega \mathrm{t} - \omega)t] + J_2(\beta)[\sin(\Omega \mathrm{t} + 2\omega)t \\ + \sin(\Omega \mathrm{t} - 2\omega)t] + J_3(\beta)[\sin(\Omega \mathrm{t} + 3\omega)t \\ + \sin(\Omega \mathrm{t} - 3\omega)t] + J_4(\beta)[\sin(\Omega \mathrm{t} + 4\omega)t \\ + \sin(\Omega \mathrm{t} - 4\omega)t] + J_5(\beta)[\sin(\Omega \mathrm{t} + 5\omega)\, \mathrm{t} \\ + \sin(\Omega \mathrm{t} - 5\omega)t] + \cdots \quad \} \end{aligned} \tag{4.15}$$

Pokud vzorec rozepíšeme pomocí goniometrických vzorců, poté aplikujeme rozklady uvedené ve vzorcích 4.12 a 4.13, kde $J_0(ß)$, $J_1(ß)$, $J_2(ß)$, $J_3(ß)$, …, $J_k(ß)$ označují Besselovy funkce prvního druhu a k-tého řádu, ty nabývají hodnot zhruba <-0.4; +1>, a rozklad 4.14, získáme výsledný vztah 4.15. Z tohoto vztahu již je evidentní obsah frekvenčního spektra. To bude obsahovat frekvence:

Ω, $\Omega + \omega$, $\Omega - \omega$, $\Omega + 2\omega$, $\Omega - 2\omega$, $\Omega + 3\omega$, $\Omega - 3\omega$, $\Omega + 4\omega$, $\Omega - 4\omega$, $\Omega + 5\omega$, $\Omega - 5\omega$,...

Celková šířka pásma frekvenční modulace vychází z tzv. Carsonova pravidla. To zní, že šířka pásma je rovna dvojnásobku součtu frekvenčního zdvihu a modulační frekvence.

5 Virtual Studio Technology

Kapitola vychází z informačního zdroje [22] a z dokumentace k VST3 pluginů distribuované spolu s Steinberg VST3 SDK.

V klasických nahrávacích studiích syntezátory, MIDI moduly, samplery a bicí automaty byly samostatné zařízení nebo zařízení zabudované do racku, ale v době výkonných počítačů si můžeme vybrat z nemalého množství softwarových hudebních nástrojů, které mohou být ovládány přes spoustu populárních hudebních programů a MIDI sekvencerů. Jakmile se staly výkonné osobní počítače snadno dostupné, každý počítač se mohl stát virtuálním hudebním studiem.

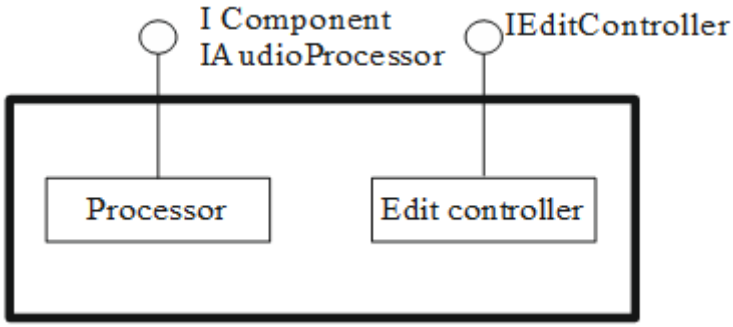
Pro uživatele MIDI sekvencerů se žádný velký pokrok nekonal až do té doby, kdy německá firma Steinberg vyvinula technologii Virtual Technology Studio, která byla zahrnuta do jejich sekvenceru Cubase, který z počátku podporoval pouze VST efekty, druhá verze označována jako VST2 již zahrnovala VST hudební nástroje.

Steinberg velmi rychle otevřel jejich VST protokol vývojářům třetí strany a umožnili tak různým softwarovým firmám vytvářet nové efekty a hudební nástroje, které mohly být použity s programem Cubase VST. Postupem času byla technologie rozšířena do většiny programů určených pro nahrávání hudby, včetně řady nekomerčních. To přiblížilo každý osobní počítač blíže k hudebnímu studiu.

V roce 2011 byla uvolněna verze třetí verze s názvem VST3, která plugin rozděluje na dvě části – část, která zpracovává nebo generuje hudební signál a část, která se zabývá sběrem informací ze vstupů a generuje uživatelské rozhraní. Tyto dvě části mohou běžet na dvou různých počítačích. VST3 také umožňuje distribuci více pluginů v jednom souboru.

5.1 Struktura VST pluginů

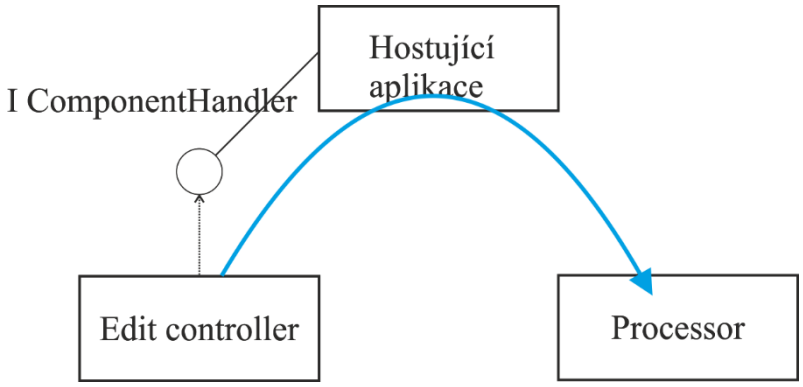
VST3 pluginy se na rozdíl od předchozích verzí liší tím, že umožňuje rozdělení na dvě části – na část zpracování (procesor) a část zpracující parametry s aktualizací grafického rozhraní (controller).



Obrázek 5.1 – Blokové schéma VST3 pluginu

Tohle rozdělení vyžaduje implementací dodatečných komponent na úrovni VST3 hosta pro spolupráci obou částí. Tato separace umožňuje spouštění každé z komponent v různém kontextu. Každá může běžet na různých počítačích, což umožňuje například používání pluginu prostřednictvím místní sítě LAN.

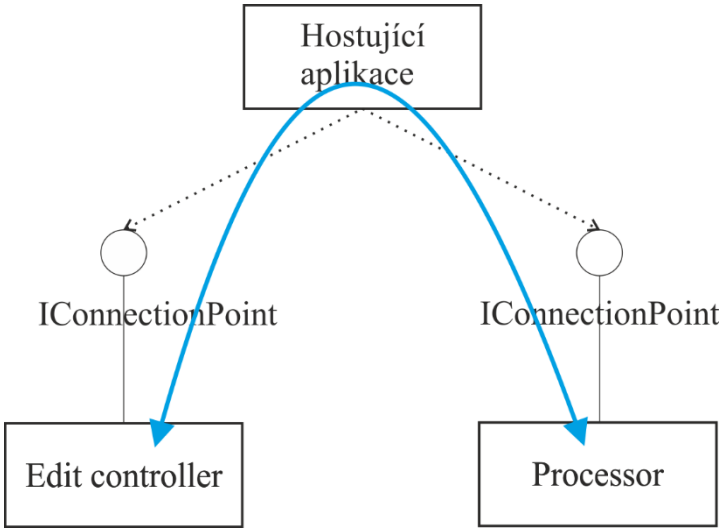
Komunikaci mezi jednotlivými komponenty zajišťuje VST host. Jsou dva typy komunikací – standardní, kdy jsou přenášeny již předem definované zprávy, jako různé změny parametrů a komunikace soukromá, kdy jsou přenášeny programátorem definované zprávy. Příkladem soukromé komunikace může být například obsah textového pole v grafickém rozhraní, jehož hodnotu ovlivňuje procesor.



Obrázek 5.2 – Standartní komunikace mezi částmi

Pomocí standardní komunikace se přenášejí definované zprávy, jako jsou změny parametrů nebo události na vstupu. Vzhledem k tomu, že se jedná o domluvenou komunikaci, může jí řídit VST host. Ten může nastavovat kompletní stavy procesoru. K tomu se využívají metody `IComponent::setState`, `IComponent::getState` a `IEditController::setComponentState`. To je potřebné hlavně tehdy, kdy je potřeba uložit a znovu načíst konfiguraci pluginu.

Dalším možným způsobem komunikace je komunikace soukromá. Procesor i controller mají domluvený význam jednotlivých zpráv, ale VST host o významu neví.

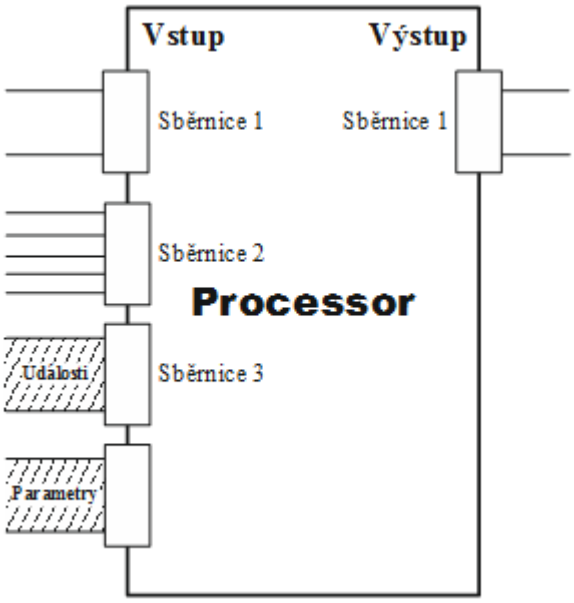


Obrázek 5.3 – Soukromá komunikace

V rámci soukromé si mohou dva komponenty spojené pomocí rozhraní `IConnectionPoint` posílat zprávy. Každá zpráva může mít svoje parametry. Procesor by neměl posílat controlleru zprávy během zpracovávání signálu. To by způsobilo zpoždění při zpracovávání signálu, které by nemohlo být realizováno v reálném čase. V takovém případě by operace odeslání zprávy měla být řešena v dalším vlákně.

5.2 Procesor

Část, která se zabývá zpracováním vstupu, se nazývá procesor nebo také processing part.



Obrázek 5.4 – Blokové schéma procesoru

Tato část může být dále rozdělena na dvě části – na entitní část a část zpracovávající signál. Důvodem rozdělení je zejména výhled do budoucna – všechny VST pluginy se sice v současnosti zabývají zpracováním zvukového signálu, firma Steinberg počítá do budoucna i s dalšími multimediálními typy.

Komponent implementuje rozhraní `IComponent`. Tato část je modelová reprezentace VST pluginu. Mimo parametry, které může controller ovlivňovat také definici sběrnice. Sběrnice je v kontextu VST pluginů funkční blok, který sdružuje datové kanály, se kterými plugin pracuje – mimo vstupní, výstupní také sběrnice událostí a parametrů. V této fázi musejí být také definovány vztahy mezi těmito sběrnicemi.

Audio procesor je klíčová část procesoru, která zpracovává zvukový signál. Implementuje rozhraní `IAudioProcessor`. Před samotným začátkem zpracování musí být procesor nakonfigurován. Konfigurace je možná pouze tehdy, když je procesor neaktivní (nic ještě nezpracovává). Nastavujeme jednotlivé parametry sběrnice, definované v entitní části.

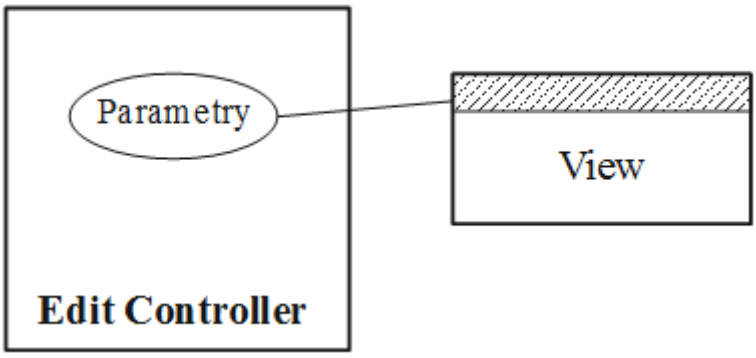
Poté následuje samotné zpracování. To je implementováno ve funkci `process` a veškerá data jsou předávány pomocí parametru `ProcessData`. Zpracovávání probíhá v blocích o určité velikosti, která může být pokaždé jiná. Tyto bloky jsou předávány jako argument `ProcessData`. Jakmile je jeden blok zpracován, zavolá se metoda `process` s dalším blokem dat.

Zvuková data, která ukládáme, jsou uloženy v bufferu, které jsou obsaženy v argumentu `ProcessData`. Obsah a struktura bufferu odpovídá definici zvukových sběrnic, včetně těch s událostmi, tudíž každá definovaná sběrnice a všechny její kanály mají svůj buffer, včetně těch deaktivovaných. Prakticky se jedná o pole float čísel, které nesou okamžitou hodnotu zvukového signálu. Data v bufferech nikdy nemohou nabýt hodnoty `NULL` (zejména výstupní buffer). VST host však může zavolat funkci `process` bez vstupních bufferů, informace určující počet vstupních vzorků či počet vstupů jsou rovny nule, což obecně signalizuje změnu parametrů. V takovém případě nemá smysl spouštět zpracování dat, ale pouze zpracovat tuto změnu parametru a běh funkce ukončit.

`ProcessData` dále obsahují informace o změnách parametrů a automatizaci – to je však záležitostí controlleru a podrobněji je popisuji níže. Obsahem této struktury jsou jaké užitečné informace, které poskytuje VST host a týkají se například vzorkovací frekvence, tempo v úderech za minutu a podobně.

5.3 Controller

Edit controller je odpovědný za definici, zpracování parametrů a jejich zanesení do uživatelského rozhraní.



Obrázek 5.5 – Blokové schéma controlleru

5.3.1 Grafické rozhraní

Controller může definovat uživatelské rozhraní, není to však povinností. Pro změnu parametrů to obecně není třeba a implementovat uživatelského rozhraní je vhodné právě tehdy, kdy je více parametrů a grafické rozhraní pomůže snadnější orientaci v nich, nebo kvůli celkovému dojmu z aplikace.

K dispozici máme několik předdefinovaných komponent s tím, že si sami můžeme vzhled komponentů vytvářet, aby komponenty ladily k celkovému dojmu z aplikace.

Grafické rozhraní může být definováno přímo v controlleru, efektivnější a přehlednější varianta je definovat grafické rozhraní ve vlastní třídě s tím, že v controlleru vytvoříme vazbu na tuto třídu.

5.3.2 Parametry

V controlleru se definují veškeré parametry, se kterými se při výpočtech pracuje. Může jich být až 2^{32} parametrů, protože každý parametr má přiřazený 32 bitový identifikátor. Hodnoty jsou vždy přenášeny jako normalizovaná reprezentace [0.0,1.0]. Pro přehlednost v rámci uživatelského rozhraní je třeba transformovat toto zobrazení do přehlednější formy.

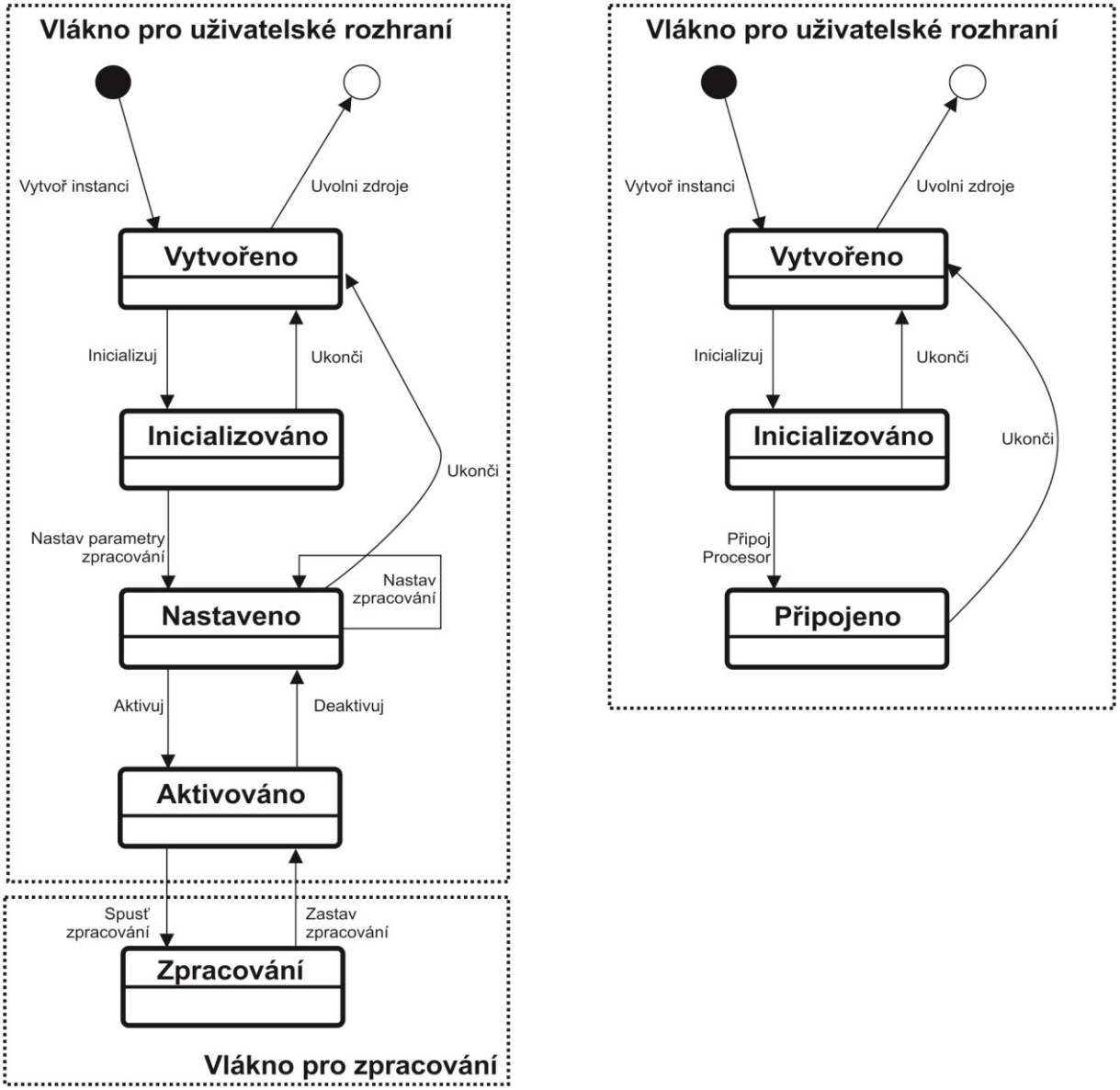
Definice parametru umožňuje určit počet kroků, ve kterých se hodnota mění. Sémantika je následující:

- je-li počet kroků roven 0, potom má parametr svoje vlastní mapování – neexistují totiž kroky mezi jednotlivými možnými hodnotami (např.: 0,454235 => 45% apod.),
- je-li počet kroků roven nějakému jinému přirozenému číslu n, počet kroků je n+1, hodnota každého kroku je 1/n.

K jednotlivým parametrům se obvykle připojují `ParameterFlags`, což určuje dodatečné vlastnosti parametru. Jedním z nich je možnost automatizace – automatická změna parametrů v určeném čase na úrovni VST hosta.

5.4 Komunikace VST hosta s VST pluginem

VST host při svém spuštění prohledá určenou složku s VST pluginy[1]. Tato složka může být určena v nastavení každého VST hosta. VST host pak prochází všechny soubory v této složce a pomocí „magického čísla“ dll souboru zjistí, jestli se jedná o VST plugin. V případě, že ano, načte si z něj základní informace o něm, aby mohl být později spuštěn.



Obrázek 5.5 – Stavový diagram při komunikaci VST hosta s VST pluginem

[1] VST3 pluginy mají výchozí cestu „`C:\Program Files (x86)\Common Files\VST3\Steinberg\`“

Sekvence vytvoření procesoru probíhá v těchto fázích:

1. **Vytvoření / zrušení instance** – v první fázi se vytvoří instance pluginu v paměti. Během této fáze máme přístup k základním informacemi o pluginu, jako je jeho název, informace o jeho tvůrci a podobně. Také obsahuje informaci o classid controlleru.
2. **Inicializace** – v rámci inicializace se provede nastavení sběrnic, jejichž počet a parametry jsou předány VST hostovi.
3. **Nastavení zpracování** – během této fáze mohou být blíže nastaveny parametry zpracování, jako například nastavení vzorkovací frekvence, dodatečné nastavení sběrnic, atd. Většina těchto parametrů nemůže být změněna během samotného zpracování, takže pokud dojde k požadavku na změny těchto parametru, musí procesor přejít ze stavu zpracování do tohoto stavu.
4. **Aktivace / deaktivace** – procesor byl aktivován a zpracování může být zahájeno. Tato fáze je přechodová a je užitečná tehdy, kdybychom chtěli udělat nějaký mezikrok mezi změnou parametru a začátkem zpracování (uložení parametrů,…).
5. **Zpracování** – v této poslední fázi již probíhá samotné zpracování signálu.

Spuštění i zastavení zpracování (přechod mezi fází 5 a 4) může být proveden z vlákna pro zpracování, musí být tedy bez zámku.

Vytvoření controlleru je snadnější a vyžaduje tyto kroky:

- **Vytvoření / zrušení instance**
- **Inicializace** – během fáze inicializace jsou definovány parametry, se kterými plugin pracuje.
- **Připojení procesoru** – v této fázi je controller připojený na jinou objektovou instanci – na příslušné uživatelské rozhraní a na procesor.

6 Implementace VST pluginu FM Synth

Zdrojem pro vzorec k výpočtu frekvence noty vychází z informačního zdroje [23].

Cílem této kapitoly je popsat implementaci VST3 pluginu, který na základě parametrů metodou frekvenční modulace generuje zvuk v reálném čase.

6.1 Vstupní bod do pluginu

Část kódu umístěná v souboru `Entry.cpp` obsahuje základní informace pro vytvoření instance pluginu a jeho částí. Výhodou je možnost distribuce více pluginů v jednom souboru – máme více procesorů a ty mají svoje controllery. Další možností je také dva různé controllery pro jeden procesor – například verze pluginu s nebo bez uživatelského rozhraní.

Na začátku se nachází makro `BEGIN_FACTORY_DEF`, které obsahuje informaci o vývojáři – mimo jeho jméno můžeme také uvést webovou adresu a emailovou adresu, včetně prefixu `mailto`.

Následují definice jednotlivých třídních instancí. Ty se uvádějí do jednoho z maker `DEF_CLASS`, `DEF_CLASS1`, `DEF_CLASS2`, `DEF_CLASS_W`. Tyto makra se liší podle jednotlivých parametrů, protože častokrát není nutné uvádět všechny. Parametry mohou být:

- **Class id –** jednoznačný 16 bytový globální jednoznačný identifikátor,
- **kardinalita třídy,**
- **kategorie pluginu –** určuje, jaké rozhraní bude host používat při instanci této třídy – jestli se jedná o controller nebo procesor,
- **jméno pluginu** – název pluginu, který bude viditelný pro uživatele v hostovi,
- **třídní příznaky** – další informace o třídě, například, jestli může běžet vzdáleně (na dvou různých počítačích), apod.,
- **podkategorie –** informace pro hosta o podkategorii pluginu – umožňuje přehlednější uspořádání nabídky VST pluginů,
- **verze pluginu**,
- **verze SDK,**
- **odkaz na konstruktor třídy.**

Parametry classid, kardinalita, kategorie, název a odkaz na konstruktor musejí být uvedeny vždy.

6.2 Controller

Definice controlleru se nachází v souboru Controler.cpp. Obsahuje nastavení parametrů a specifikaci grafického rozhraní, které je implementováno ve zvláštní třídě.

6.2.1 Definice parametrů

Plugin bude pracovat s následujícími parametry:

- **Hlasitost (amplituda) celkového signálu.**
- **Vyvážení celkového signálu.**
- **Attack** – doba v milisekundách určující, za jak dlouho vzroste amplituda z nuly do požadované hlasitosti. Obor hodnot je od 1ms do 1000ms.
- **Hold** – doba, jak dlouho bude tón znít předtím, než nastane fáze release. Maximální hodnota, kterou může parametr nabývat, je nekonečno – v tomto případě tón zní až do uvolnění klávesy.
- **Release** – doba v milisekundách určující, za jak dlouho klesne amplituda z požadované hlasitosti na nulu poté, co uplyne doba Hold nebo je uvolněna klávesa.
- **Mód Attack** – určení řady, kterou bude signál stoupat – k dispozici je lineární, exponenciální a logaritmická řada.
- **Mód Release.**
- **Pitch pro oscilátor 1 až 4** – odchylka od jmenovité frekvence příslušného tónu maximálně ± 2 půltóny.
- **Násobek frekvence modulátoru pro modulaci nosné pro oscilátor 1 až 4.**
- **Typ modulační vlny pro oscilátor 1 až 4** – podporované typy vln jsou sinus, trojúhelníkový, pilový a obdélníkový signál
- **Hlasitost oscilátoru 1 až 4.**
- **Vyvážení oscilátoru 1 až 4.**
- **Příznak pro aktivaci/deaktivaci oscilátoru 1 až 4.**

Každý parametr má svůj jednoznačný, celočíselný identifikátor. K usnadnění přehlednosti při používání těchto identifikátorů používáme enum `ParameterIDs`. Vlivem jednotlivých parametrů na generovaný signál se podrobněji zabývám v kapitole 6.3.

6.2.2 Mapování MIDI událostí na parametry

Mimo samotné noty se jako událostí může přenášet spousta jiných událostí, které nesou také hodnotu. Za pomocí těchto událostí můžeme měnit výše uvedené parametry. Pro účel pluginu FM synth mapujeme výše uvedené parametry hlasitost, vyvážení, doby Attack a Release.

6.2.3 Grafické rozhraní

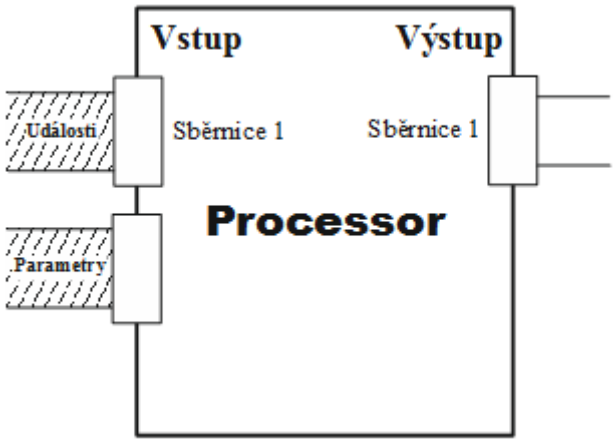
Grafické rozhraní je implementováno v souboru GUI.cpp. Tato třída obsahuje funkcionality pro reprezentaci grafického rozhraní a pro příjem či odesílání změn parametru. V konstruktoru je nutné nastavit, jak často se budou aktualizovat hodnoty jednotlivých komponentů.

Součástí grafického rozhraní je komponenta „stavový řádek“, která je umístěna mezi hlasitostí a vyvážením. Obsahem této komponenty je vždy hodnota právě měněného parametru. To nabízí značnou úsporu prostoru v rámci okna GUI.

6.3 Procesor

Procesor realizuje samotnou syntézu. Plugin bude potřebovat dvě sběrnice – vstupní, kterou budou tvořit události a výstupní, na kterou budeme posílat vygenerovaný signál.

Na vstupu jsou události, což je stlačení a uvolnění klávesy. Jiné vstupní události nepotřebujeme. Sběrnice s parametry zase informuje procesor o změnách parametru. Obě sběrnice jsou obousměrné. Výstupní sběrnice je audio sběrnice a obsahuje dva kanály – je tedy stereofonní.



Obrázek 6.1 – Blokové schéma procesoru pluginu FM synth

Samotné zpracování má potom tři fáze:

- **Obsluha změny vstupních parametrů** – pokud se změní některý z parametrů, controller procesor o tom informuje. Veškeré změny zachytíme a uložíme do entitní části procesoru.
- **Načtení událostí ze sběrnice události** – vstupní sběrnice s událostmi obsahuje informace o stlačené notě a o jejím uvolnění. Pokud se jedná o stlačení, uložíme informace o ní do struktury, která se později prochází a ze které se generují frekvence. Pokud se jedná o uvolnění noty, informujeme strukturu o tom, že nota má být smazána poté, co uběhne parametr „release“.
- **Výpočty** – samotné výpočty, které generují zvuk v závislosti na vstupních parametrech.

U některých pluginů by mohla být obsažena ještě čtvrtá část, která by se starala o obsluhu změny výstupních parametrů – to by nastávalo v případě, že bychom během výpočtu chtěli některý z parametrů změnit a controller by o tom prostřednictvím této fáze byl informován.

Jak již bylo uvedeno v přecházející kapitole, veškeré zpracování probíhá v metodě `process` a data jsou předána odkazem prostřednictví proměnné `processData`.

6.3.1 Načtení událostí ze vstupu

Pokud nastane nějaká změna některého z parametrů, je třeba tuto změnu zaznamenat. V případě, že žádná změna neproběhla, můžeme tuto část zpracování přeskočit.

Veškeré změny jsou uloženy v struktuře, kterou můžeme sekvenčně procházet. Každý parametr má svůj jednoznačný celočíselný identifikátor, podle kterého poznáme, o změnu kterého parametru se jedná. Z důvodu vyšší přehlednosti se používá enum. Po identifikaci následuje uložení do entitní části procesorů. Jelikož se změny parametru přenáší v normalizovaném tvaru, musíme tyto hodnoty ještě převést do požadovaného tvaru. Přehlednost kódu můžeme zvýšit tím, že k jednotlivým výpočtům použijeme makro – u některých parametrů se výpočty mohou opakovat.

Poté následuje načtení ze vstupu. Algoritmus načtení změny je naprosto stejný, jako u načtení parametrů a jejich změn. V tuto chvíli odchytáváme dvě události: `NoteOn` událost a `NoteOff` – přičemž jedna signalizuje stlačení noty a druhá uvolnění noty. Poté už následuje zpracování zvukového signálu.

6.3.2 Zpracování zvukového signálu

Ještě před jakýmkoliv zpracováním signálu musíme vytvořit datovou strukturu, která bude obsahovat veškeré noty a hlavně údaje o jejich frekvenci.

$$f = f_r \left(\frac{1}{12}\right)^k \tag{6.1}$$

Vzorec 6.1 definuje, jak vypočítat frekvenci kterékoliv noty. Kde f_r je referenční frekvence (obvykle 440 Hz) a kde k je počet půltónu od frekvence referenční.

VST umožňuje generovat frekvence od C_{-2} až do G_8. Tóny pod C_0 jsou již téměř u hranice slyšitelnosti a navíc po uplatnění frekvenční modulace při nízkých kmitočtech se uplatňuje efekt vibrato, proto možný rozsah je od C_0 do G_8.

Informace o příslušné notě bude také obsahovat počet již vygenerovaných vzorků, což slouží k tvarování vlny z hlediska obálky. Důležitý je také údaj, který obsahuje informaci o fázi, ve které se tón vzniklý z noty nachází – jestli ve fázi zesílení, zeslabení či jestli je aktivní nebo neaktivní.

V rámci zpracování vstupních události přidáváme či odebíráme noty z další struktury, která určuje, které noty se budou generovat. Entita každé `NoteOn` či `NoteOff` události obsahuje základní informace o notě – její pořadí (přičemž C_{-2} je 0, G_8 je 127), sílu stlačení, jednoznačný identifikátor a podobně. Nicméně tento identifikátor nemusí být vždy k dispozici, proto je třeba brát v úvahu jako jednoznačný identifikátor pořadí noty – může se sice opakovat, ale v jeden okamžik hraje nota pouze jednou.

Následuje výpočet hodnoty modulačního kmitočtu, což záleží na konfiguraci příslušných oscilátorů, které nabízejí čtyři signály – klasický sinusový, obdélníkový, trojúhelníkový a pilový signál. Všechny tyto odvozené signály se dají zapsat pomocí Fourierovy řady jako součet několika sinusovek, nicméně takové řešení by při běhu všech čtyř oscilátorů bylo nevýhodné. Všechny tyto signály lze zapsat jednodušeji, aniž by se změnila barva zvuku.

$$y(t) = \begin{cases} +1 \Leftrightarrow t < \frac{1}{2}T \\ 0 \Leftrightarrow t = 0 \\ -1 \Leftrightarrow t > \frac{1}{2}T \end{cases} \tag{6.2}$$

$$y(t) = 1 - 2\frac{t}{T} \tag{6.3}$$

$$y(t) = \begin{cases} \frac{4\,t}{T} & pro\ \frac{t}{T} < \frac{1}{4} \\ 1 - 4\left(\frac{t}{T} - \frac{1}{4}\right) & pro\ \frac{t}{T} > \frac{1}{4}\ a\ \frac{t}{T} < \frac{1}{2} \\ -4\left(\frac{t}{T} - \frac{1}{2}\right) & pro\ \frac{t}{T} > \frac{1}{2}\ a\ \frac{t}{T} < \frac{3}{4} \\ -1 + 4\left(\frac{t}{T} - \frac{3}{4}\right) & pro\ \frac{t}{T} > \frac{3}{4} \end{cases} \tag{6.4}$$

Obdélníkový signál (vzorec 6.2) tvoří pouze tři hodnoty – na začátku signálu je rovna nule, v první polovině *+1* a v druhé polovině *–1*. O amplitudu se nestaráme, protože ta je určena modulačním indexem.

Pilový signál (vzorec 6.3) lineárně stoupá z hodnoty *–1* na hodnotu *+1*. Zjednodušeně lze říct, že hodnota na výstupu je rovna poměru okamžitě frekvenci vůči celkové frekvenci. Jelikož tento poměr nabývá kladných hodnot a pilový signál musí stoupat ze záporných hodnot, musíme odečíst jedničku a samotný poměr následně vynásobit dvěma – celkový rozsah hodnot musí být z *–1* do *+1*.

Trojúhelníkový signál (vzorec 6.4) tvoří čtyři přímky, které rostou do svých maximálních hodnot po čtvrtinách.

Po výpočtu modulačního signálu proběhne výpočet signálu nosné, ten bude vždy sinusový. V závislosti na konfiguraci oscilátoru se část signálu přesune do pravého kanálu, část do levého a sečte se s hodnotami z předešlých oscilátorů. Jedná se v podstatě o aditivní syntézu, tudíž by celková hodnota oscilátorů měla být vydělena jejich počtem, aby se amplituda nezvyšovala do vysokých hodnot. To se ovšem netýká při sčítání jednotlivých tónů – určitý vzrůst amplitudy je u většiny syntezátorů běžné.

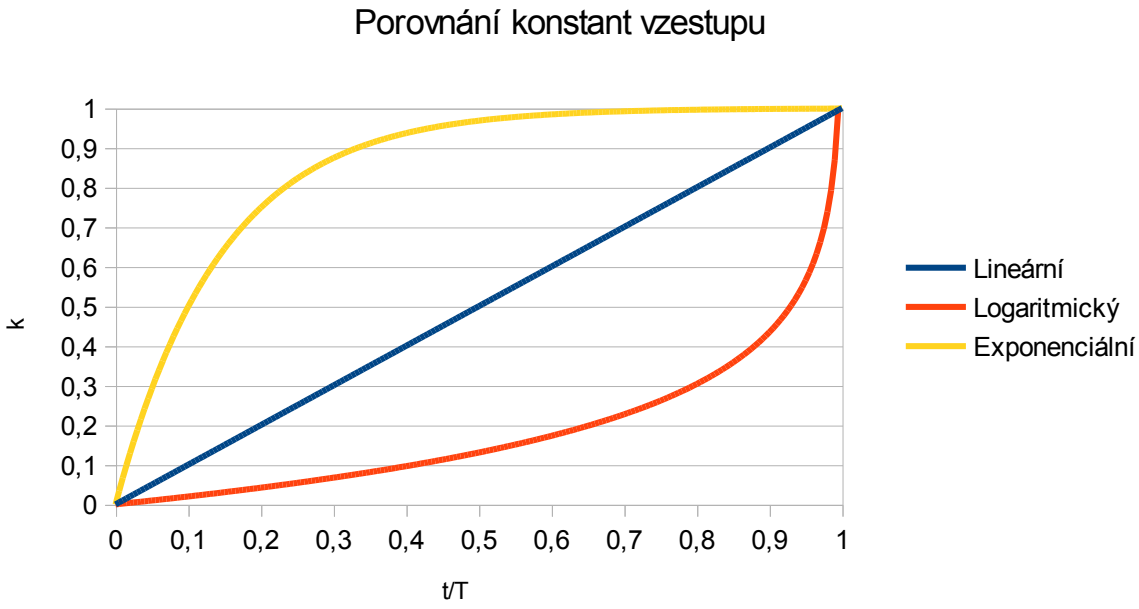
V další fázi musí být vytvarována obálka signálu. K dispozici je ASR obálka. Pokud se signál nachází ve fázi náběhu, bude z nuly stoupat do hodnoty amplitudy signálu. Uživatel může zvolit, zdali bude signál stoupat lineárně, exponenciálně či logaritmicky. Je žádoucí třeba minimální hodnota doby náběhu, protože skoková změna z nuly na nenulovou hodnotu způsobuje rušivé zvuky.

$$k = \frac{t}{T} \tag{6.5}$$

$$k = log_{20}\left(\frac{t}{T}\right) \tag{6.6}$$

$$k = 1 - 0{,}001^{\frac{t}{T}} \tag{6.7}$$

Lineární průběh (vzorec 6.5) je v podstatě intuitivní – dělíme okamžitý časový okamžik vůči celkovému časovému okamžiku. V případě logaritmu (vzorec 6.6) musíme vhodně vybrat řád logaritmu tak, aby hodnota členu v případě konce doby náběhu byla maximálně jedničková. Totéž platí i v případě exponenciálního průběhu (vzorec 6.7).



Obrázek 6.2 – Porovnání konstant vzestupu

Jak je patrné z obrázku 6.2 exponenciální průběh stoupá nejrychleji a skoro polovinu doby trvání se nachází na hodnotě amplitudy signálu. Kdežto logaritmický průběh stoupá mnohem pomaleji, většinu doby trvání se amplituda signálu nachází v polovině jmenovité a během necelé desetiny stoupne na dvojnásobek.

Pokud se hodnota konstanty vzrůstu rovná hodnotě větší, než jedna, signál pokračuje do fáze trvání. Ta trvá v standardní konfiguraci nekonečně dlouho (tudíž do uvolnění klávesy), nebo určený časový okamžik, po jehož uplynutí bude automaticky vyvolána následující fáze – uvolnění. V této fázi signál zní nadále určenou dobu s tím, že jeho amplituda klesá k nule.

$$k = 1 - \frac{t_r}{T_r} \tag{6.8}$$

$$k = -log_{20}\left(1 - \frac{t_R}{T_R}\right) \tag{6.9}$$

$$k = 1 - 0{,}001^{\frac{t_r}{T_r}} \tag{6.10}$$

Jelikož doba přehrání příslušného tónu skončila, musíme pracovat s časovými okamžiky vyhrazenými pro fázi uvolnění.

Po vytvarování obálky následuje modifikace amplitudy signálu v závislosti na nastavené hlasitosti a vyvážení signálu mezi pravým a levým kanálem v rámci nastaveného vyvážení. Poté již je zvukový signál odeslán na výstup a zpracování pokračuje.

7 Závěr

V rámci této bakalářské práce se čtenář mohl seznámit s metodami digitální syntézy zvuku a s jejich implementací do technologie VST3.

Práce obsahovala šest částí, v první části byly uvedeny základy akustiky a digitální reprezentace zvuku. Druhá část byla zaměřena na digitální syntézu zvuku – byly popsány její nejčastější způsoby, zařízení jí realizující a nejčastější DSP metody. Hlavním účelem této kapitoly bylo vysvětlit metody, na základě kterých VST pluginy pracují.

Třetí část se věnovala frekvenční modulaci a o jejím vlivu na barvu tónu. Tato metoda digitální syntézy byla využita v praktické části práce, v pluginu FM Synth. Čtvrtá část popisovala VST3 pluginy z její technické části. Rozebrány byly jednotlivé části – část, která zpracovává zvukový signál a část, která se stará o změnu parametrů a jejich zanesení do uživatelského rozhraní. Popsána také byla komunikace mezi těmito částmi.

Pátá část se věnovala implementaci jednoduchého VST3 pluginu s grafickým uživatelským rozhraním. Postup byl podrobně rozepsán a vysvětleny byly veškeré důležité algoritmy, které v pluginu jsou obsaženy.

Praktická část bakalářské práce byla prezentována jako implementace VST3 pluginu. Tento plugin zahrnuje základní možnosti technologie. Do budoucna by plugin mohl být doplněn do další parametry, jako například obálka pro každý oscilátor nebo různé efekty, jako například vibrato, ozvěny a podobně.

Stejně tak se počítá i s rozvojem VST3 hostujících sekvencerů. V současnosti tuto nejnovější verzi VST protokolu podporuje jenom zlomek z nich a většinou pouze v testovacím režimu. Tato nízká podpora odrazuje většinu programátorů od vývoji pluginů v této verzi, což zase znemožňuje vyhledání a následnou opravu chyb v SDK. Potenciál třetí generace VST pluginů zůstává zapovězen. Uběhne asi ještě dlouhá doba, než obliba této generace přiblíží generaci druhé a stane se standardem v rámci VST protokolu.

Použitá literatura

[1] VŠETEČKA, Martin a Jaroslav REICHL. *Encyklopedie Fyziky* [online]. 2006 [cit. 2013-04-27]. Dostupné z: http://fyzika.jreichl.com/

[2] *Pulse-code modulation*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://en.wikipedia.org/wiki/Pulse-code_modulation

[3] *Zvuk*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://cs.wikipedia.org/wiki/Zvuk

[4] *Akustika, vznik a šíření zvuku, frekvenční analýza a syntéza, sluchový vjem zvukového signálu*. In: BERNAT, Petr. Anatomie varhan [online]. 2010 [cit. 2013-04-23]. Dostupné z: http://homen.vsb.cz/~ber30/texty/varhany/anatomie/pistaly_akustika.htm

[5] ROADS, Curtis. *The computer music tutorial*. Cambridge, Mass.: MIT Press, c1996, xx, 1234 p. ISBN 02-626-8082-3.

[6] *Digital Signal Processing Central* [online]. 2013 [cit. 2013-04-19]. Dostupné z: dspguru.com/

[7] MARSHALL, David. *Digital Audio Synthesis*. In: [online]. [cit. 2013-04-06]. Dostupné z: http://www.cs.cf.ac.uk/Dave/Multimedia/PDF/04_CM0340_Synthesis.pdf

[8] Additive synthesis. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://en.wikipedia.org/wiki/ Additive_synthesis

[9] *Metody zvukové syntézy*. In: Metody zvukové syntézy [online]. 2010 [cit. 2013-04-27]. Dostupné z: http://elektronicka-hudba.telotone.cz/clanky/metody-zvukove-syntezy/

[10] *Synthesizer*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://en.wikipedia.org/wiki/ Synthesizer

[11] *Digital Signal Processing Central* [online]. 2013 [cit. 2013-04-19]. Dostupné z: dspguru.com/

[12] JOHN STRAWN, Editor a With contributions by F. Richard Moore ... [et]. AL]. *Digital audio signal processing: an anthology*. Madison, Wis: A-R Editions, 1985. ISBN 08-957-9279-6.

[13] MARSHALL, David. Digital Audio Synthesis. In: [online]. [cit. 2013-04-06]. Dostupné z: http://www.cs.cf.ac.uk/Dave/Multimedia/PDF/04_CM0340_Synthesis.pdf

[14] ROADS, Curtis. *The computer music tutorial*. Cambridge, Mass.: MIT Press, c1996, xx, 1234 p. ISBN 02-626-8082-3.

[15] MAUTNER, Pavel. Diskrétní Fourierova transformace. In: Pavel Mautner [online]. 2012 [cit. 2013-04-27]. Dostupné z: www.kiv.zcu.cz/~mautner/zvi/fourier/ZVI_Fourierova_transformace.ppt

[16] Finite impulse response. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://en.wikipedia.org/wiki/ Finite_impulse_response

[17] Infinite impulse response. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://en.wikipedia.org/wiki/ Infinite_impulse_response

[18] Digitální signálový procesor. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-27]. Dostupné z: http://cs.wikipedia.org/wiki/ Digitální_signálový_procesor

[19] Odvození vztahu pro šířku pásma FM. In: Elektřina a magnetismus [online]. 2011 [cit. 2013-04-22]. Dostupné z: http://lucy.troja.mff.cuni.cz/~tichy/elektross/aplikace/radio/ odvozeni_fm.html

[20] REID, Gordon. An Introduction To Frequency Modulation. [online]. [cit. 2013-03-30]. Dostupné z: http://www.soundonsound.com/sos/apr00/articles/synthsecrets.htm

[21] REID, Gordon. More On Frequency Modulation. [online]. [cit. 2013-03-30]. Dostupné z: http://www.soundonsound.com/sos/may00/articles/synth.htm

[22] WHITE, Paul. Basic VST instruments. London: Sanctuary, 2001. ISBN 18-607-43609.

[23] SUITS, Bryan. Equations for the Frequency Table. In: *Physics Department, MTU* [online]. 2002 [cit. 2013-04-30]. Dostupné z: http://www.phy.mtu.edu/~suits/

Seznam příloh

Součástí BP je CD.

Příloha.A: Adresářová struktura CD

- **Plugin** – adresář obsahuje výsledný VST3 plugin.
- **Samples** – zvukové ukázky vygenerované pluginem FM synth.
- **Source code** – zdrojový kód k aplikaci + VST3 SDK
- **VST host** – jednoduchý program umožňující hostování VST3 plugin